pMatlab Takes the HPCchallenge

Ryan Haney, Hahn Kim, Andrew Funk, Jeremy Kepner, Charles Rader, Albert Reuther and *Nadya Travinin*
MIT Lincoln Laboratory, Lexington, MA 02420
Phone: 781-981-2514
Email Addresses: {haney, hgk, afunk, kepner, rader, reuther, nt}@ll.mit.edu

Abstract[1]
The HPCchallenge benchmark suite has been released by the DARPA HPCS program to help define the performance boundaries of future Petascale computing systems. The suite is composed of several well known computational kernels (STREAM, Top500, FFT, and RandomAccess) that span high and low spatial and temporal locality. These kernels also encompass key aspects of embedded signal processing: vector computations, matrix multiplies, corner turns and random selection operations. MATLAB®[2] is the primary high level language used within the signal processing community and is increasingly used for large system simulations and quickly processing data in the field. The pMatlab parallel MATLAB toolbox provides the necessary global array semantics to allow HPCchallenge to be implemented. The results provide a unique opportunity to probe both the relative (pMatlab vs. MATLAB) and absolute (pMatlab vs. C/Fortran+MPI) merits of pMatlab. Specifically, for each kernel in HPCchallenge we examine code size, maximum problem size, and performance. We find pMatlab code to be approximately 10x smaller than the equivalent C/MPI code. The problem sizes possible using pMatlab scale linearly with the number of processors (e.g. we are able to FFT a 2^{28} complex vector on 16 CPUS), and are comparable to the corresponding C/Fortran+MPI code. Finally, the scalability of the kernels approaches that of the C/Fortran+MPI code.

Introduction
The HPCchallenge
The DARPA High Productivity Computing Systems (HPCS) program has initiated a fundamental reassessment of how we define and measure performance, programmability, portability, robustness and, ultimately, productivity in the HPC domain [1]. With this in mind, HPCchallenge is designed to approximately bound computations of high and low spatial and temporal locality for Petascale systems. Figure 1 illustrates the approximate spatial/temporal relationship of the different kernels and the connections to important operations in the embedded

signal processing community. In addition, because HPCchallenge consists of simple mathematical operations, this provides a unique opportunity to look at language and parallel programming model issues. This paper compares traditional C/Fortran+MPI with MATLAB using global array semantics.
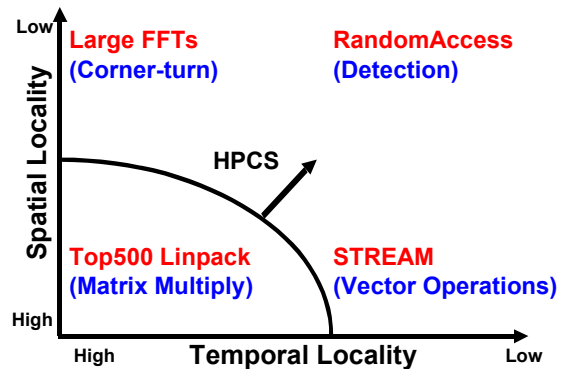

Figure 1: HPCchallenge kernels are plotted relative to spatial and temporal locality.

The pMatlab Parallel Toolbox
The pMatlab toolbox implements global array semantics in MATLAB. pMatlab provides high-level parallel data structures and functions without removing the fast prototyping capability and ease of use for which MATLAB is well known [2]. This is achieved by combining operator and function overloading with the concept of parallel data and task mapping to provide implicit data and computational parallelism. pMatlab is currently being used for simulating signal processing chains and for rapid analysis of sensor data in the field. The implementation of the HPCchallenge using pMatlab offers a means for more detailed performance analysis of pMatlab.

Parallel Implementation
STREAM consists of four local operations performed on distributed vectors: copy, scaling, addition, and scaling with addition. All of these operations are important in signal and image processing. The STREAM benchmark requires no interprocessor communication and is implemented using simple distributed matrices.

RandomAccess is designed to measure the random access capabilities of a computer system. This is accomplished by effectively computing the histogram of a random number generator, replacing the typical addition

[2] MATLAB is a registered trademark of The Mathworks, Inc.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
01 FEB 2005	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
pMatlab Takes the HPCchallenge	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
MIT Lincoln Laboratory, Lexington, MA 02420	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM00001742, HPEC-7 Volume 1, Proceedings of the Eighth Annual High Performance Embedded Computing (HPEC) Workshops, 28-30 September 2004 Volume 1., The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	**UU**	**7**	
unclassified	**unclassified**	**unclassified**			

update with a bit level XOR operation. The ability to randomly access data and perform logical operations are standard "post detection" signal processing operations. RandomAccess requires dynamic communications among all the processors and is implemented using parallel sparse arrays.

The Top500 Linpack Benchmark uses an LU Solver to solve a dense linear system of equations such as *Ax=b*. Such an algorithm requires selecting and communicating arbitrary parallel sub-matrices typical of many dense linear algebra operations. At the core of LU are matrix-matrix multiplies typical of multi-element beamforming operations.

The FFT kernel performs a 1-D Fast Fourier Transform. The 1-D FFT is performed by computing two 2-D FFTs, and then corner-turning the distributed matrix in between the two computations. Both the local 2D FFTs and large matrix corner turns are among the most important operations in multi-sensor signal processing.

Results

For each kernel in the HPCchallenge, we examine code size, maximum problem size, and performance on a Linux cluster consisting of dual 3.0 GHz Xeon processors connected with Gigabit Ethernet. Examining code size, we find pMatlab code to be approximately 10x smaller than the equivalent C/F77+MPI code. Approximate software lines of code numbers for the HPCchallenge kernels are shown in Table 1.

The maximum problem sizes possible using pMatlab scale linearly with the number of processors used and are comparable to the corresponding C/F77+MPI code. Figure 2 illustrates this for the Top500 kernel. The maximum input matrix size run on 16 processors (28K x 28K) is 16x the maximum size that can be run on a single processor (7K x 7K). Figure 3 shows the performance and maximum problem size achieved in the pMatlab FFT code relative to serial MATLAB, which uses FFTW [4] to implement its Fourier Transform. The performance scalability is typical of that seen in C/F77+MPI implementation.

Lines of code	C/F77 + MPI	pMatlab	C/F77+MPI / pMatlab
STREAM	441	51	~8
Random Access	225	101	~2
FFT	~1100	72	~15
Top500	~5000	200	~25

Table 1: C/Fortran + MPI vs. pMatlab software lines of code for four of the HPCchallenge benchmarks.
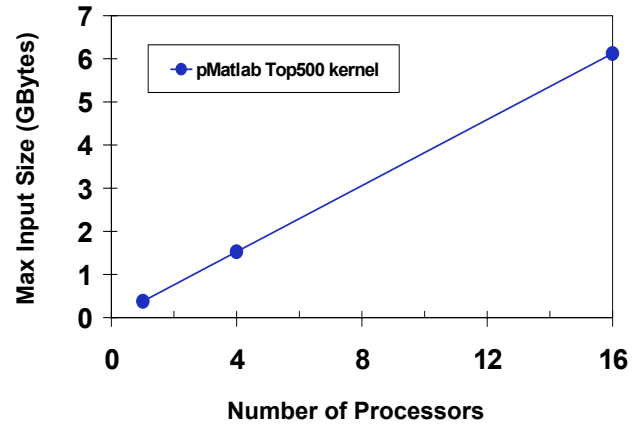
Figure 2: Maximum input matrix data sizes are plotted for the Top500 kernel. Each matrix contained real double-precision data.
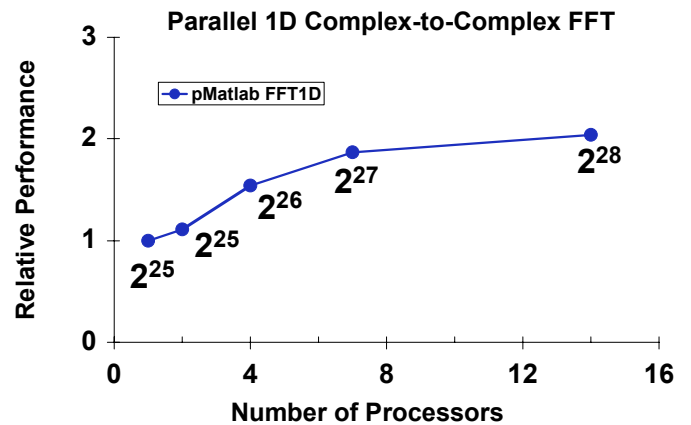
Figure 3: Performance (Flops) and scalability results are plotted for the FFT kernel. Results are relative to the serial MATLAB performance. Numbers next to the points indicate the size of the complex vector used.

References

[1] HPCS - High Productivity Computer Systems. http://www.highproductivity.org, 2004.
[2] Jeremy Kepner and Nadya Travinin. "Parallel MATLAB: The Next Generation". HPEC 2003 Workshop, 2003.
[3] Jack Dongarra. "Performance of Various Computers Using Standard Linear Equations Software". University of Tennessee, Knoxville TN. http://www.netlib.org/benchmark/performance.ps, 2004.
[4] FFTW Fastest Fourier Transform in the West. http://www.fftw.org, 2004.

pMatlab Takes the HPCchallenge

Ryan Haney, Hahn Kim, Andrew Funk, Jeremy Kepner, Charles Rader, Albert Reuther and Nadya Travinin

HPEC 2004

MIT Lincoln Laboratory

Motivation and Goals

- **Motivation**
  - **The DARPA HPCS program has created the HPCchallenge benchmark suite in an effort to redefine how we measure productivity in the HPC domain**
  - **Implementing the HPCchallenge benchmarks using pMatlab allows a unique opportunity to explore the merits of pMatlab with respect to HPEC**

- **Goals**
  - **Compare traditional C/MPI with pMatlab. Measurements of productivity include:**
    - **Maximum problem size: Largest problem that can be solved or fit into memory**
    - **Execution performance: Run-time performance of the benchmark**
    - **Code size: Software lines of code (SLOC) required to implement the benchmark**

HPCchallenge Relevance to HPEC

- **HPCchallenge benchmarks encompass key embedded signal processing operations**
    - **FFT:** Distributed corner turn and FFTs important in multi-sensor signal processing
    - **RandomAccess:** Random data accesses typical of "post detection" operations
    - **Top500:** Matrix-matrix multiplies typical of multi-element beamforming
    - **STREAM:** Distributed vector operations common to signal processing

FFT Results

Algorithm

FFT rows →   FFT columns

	C/MPI	pMatlab	Ratio
SLOC	2509	72	**35**

Software Code Size

- **pMatlab memory scalability comparable to C/MPI (128x on 128 CPUs)**

- **pMatlab execution performance comparable to C/MPI (55x on 128 CPUs)**

- **pMatlab code size is 35x smaller than C/MPI**

pMatlab Takes the HPCchallenge

Ryan Haney, Hahn Kim, Andrew Funk, Jeremy Kepner, Charles Rader, Albert Reuther, Nadya Travinin

Motivation

- The DARPA HPCS program has created the HPCchallenge benchmark suite in an effort to redefine how we measure productivity in the HPC domain

- MATLAB® is the primary high level language used within the signal processing community; increasingly used for
  - large system simulations
  - processing data in the field

- pMatlab implements global array semantics in MATLAB
  - Global array semantics allow indexing and general element access for distributed data

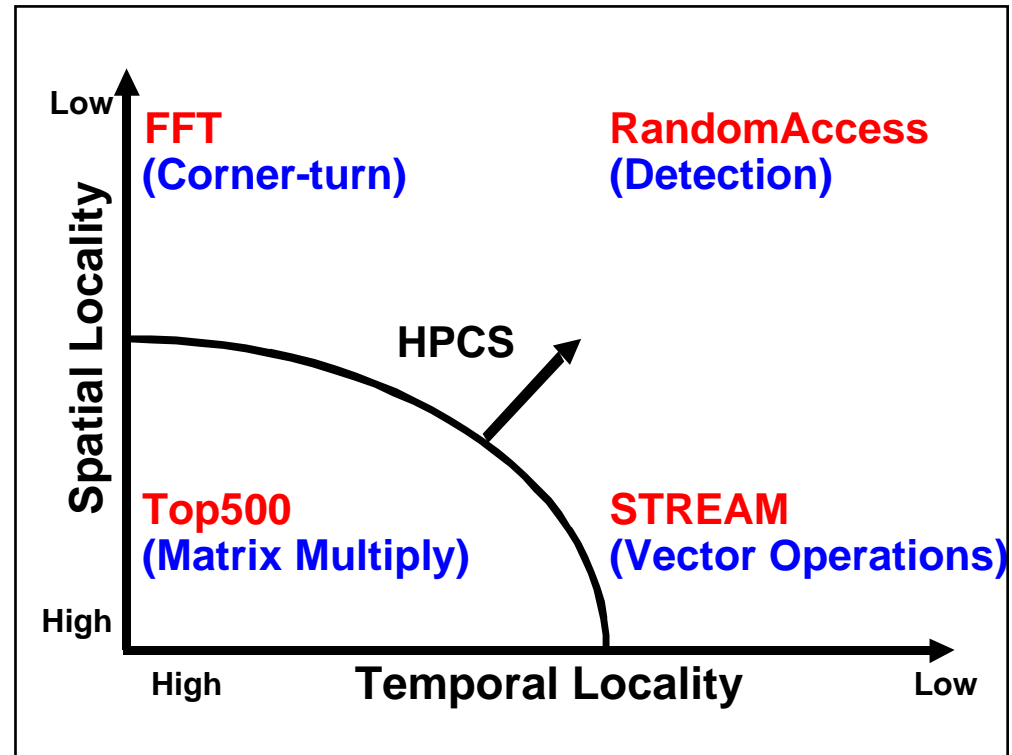- Implementing the HPCchallenge benchmarks using pMatlab allows a unique opportunity to explore the merits of pMatlab with respect to high performance embedded computing

Local Benchmarks
- DGEMM (matrix x matrix multiply)
- STREAM
  - COPY, SCALE, ADD, TRIAD
- RandomAccess
- FFT

Global Benchmarks
- Top500 (High Performance LINPACK)
- PTRANS — parallel matrix transpose
- RandomAccess
- FFT

HPCchallenge

HPCchallenge Relevance to HPEC

- Four key benchmarks have significant relevance to HPEC
  - **FFT:** Distributed corner turn and FFTs important in multi-sensor signal processing
  - **RandomAccess:** Random data accesses typical of "post detection" operations
  - **Top500:** Matrix-matrix multiplies typical of multi-element beamforming
  - **STREAM:** Distributed vector operations common to signal processing

- Multiple implementations
  - C/Fortran, C/Fortran+MPI, MATLAB, pMatlab

Goals

- Implement and analyze the performance of HPCchallenge benchmarks using pMatlab

- Optimize and add functionality to the pMatlab toolbox

- Compare traditional C/MPI with MATLAB using global array semantics. Measurements of productivity include:
  - **Maximum problem size:** Largest problem that can be solved or fit into memory
  - **Execution performance:** Run-time performance of the benchmark
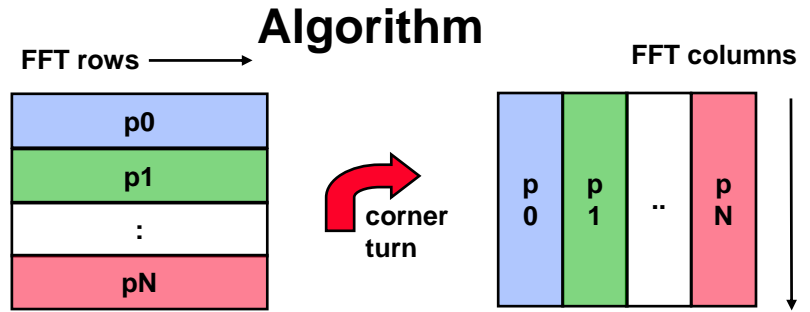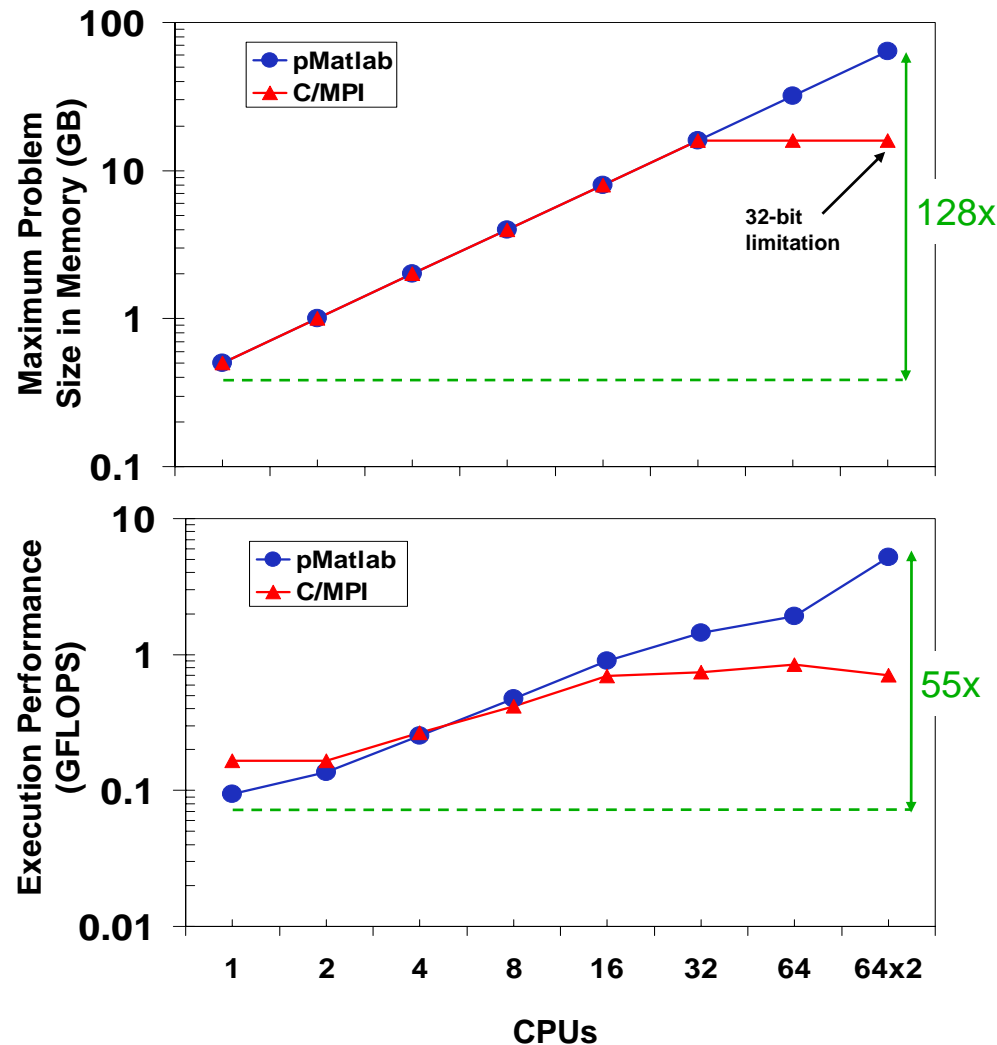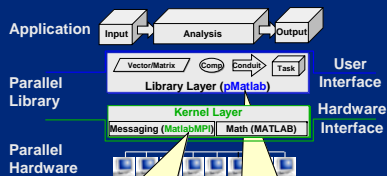 - **Code size:** Software lines of code (SLOC) required to implement the benchmark

FFT Results

Software Code Size

	C/MPI	pMatlab	Ratio
SLOC	2509	72	35

- pMatlab memory scalability comparable to C/MPI (128x on 128 CPUs)
- pMatlab execution performance comparable to C/MPI (55x on 128 CPUs)
- pMatlab code size is 35x smaller than C/MPI

RandomAccess (v0.5) Results

Software Code Size

	C/MPI	pMatlab	Ratio
SLOC	603	101	6

- pMatlab memory scalability comparable to C/MPI (128x on 128 CPUs)
- pMatlab execution performance comparable to C/MPI
- pMatlab code size is 6x smaller than the C/MPI implementation

FFT (Corner-turn) **RandomAccess (Detection)**

Spatial Locality — Low / High

HPCS

Top500 (Matrix Multiply) **STREAM (Vector Operations)**

Temporal Locality — High / Low

pMatlab
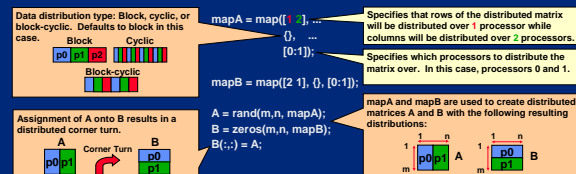
pMatlab Software Architecture

- Can build a parallel library with a few messaging primitives
- **MatlabMPI** provides this messaging capability:

```
MPI_Send(dest,comm,tag,X);
X = MPI_Recv(source,comm,tag);
```
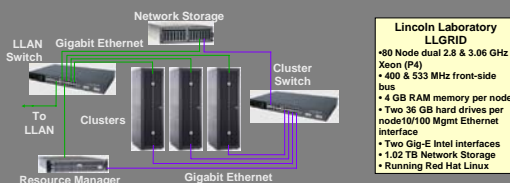
- Can build applications with a few parallel structures and functions
- **pMatlab** provides parallel arrays and functions

```
X = ones(n,mapX);
Y = zeros(n,mapY);
Y(:,:) = fft(X);
```

Top500 Results

Software Code Size

	C/MPI	pMatlab	Ratio
SLOC	15,561	235	66

- pMatlab maximum problem size scales 86x on 128 CPUs
- pMatlab execution performance scales 3x
  - Removing index calculation overhead will significantly improve
- pMatlab code size is 66x smaller than C/MPI implementation

STREAM Results

Software Code Size

	C/MPI	pMatlab	Ratio
SLOC	441	51	8

- pMatlab memory scalability comparable to C/MPI (128x on 128 CPUs)
- pMatlab execution performance comparable to C/MPI (128x on 128 CPUs)
- pMatlab code size is 8x smaller than C/MPI

Conclusions

Benchmark Results Summary

- Memory scalability comparable to C/MPI on nearly all of HPCchallenge (for 128 CPUs). Allows MATLAB users to work on much larger problems.

- Execution performance comparable to C/MPI on nearly all of HPCchallenge (for 128 CPUs). Allows MATLAB users run their programs much faster.

- Code size much smaller. Allows MATLAB users to write programs much faster than C/MPI

- pMatlab allows MATLAB users to effectively exploit parallel computing, and can achieve performance comparable to C/MPI.

pMatlab Goal: Maps and Distributed Matrices

Data distribution type: Block, cyclic, or block-cyclic. Defaults to block in this case.

mapA = map([1 2], ...
{}, ...
[0:1]);

Specifies that rows of the distributed matrix will be distributed over 1 processor while columns will be distributed over 2 processors.

mapB = map([2 1], {}, [0:1]);

Specifies which processors to distribute the matrix over. In this case, processors 0 and 1.

Assignment of A onto B results in a distributed corner turn.

A = rand(m,n, mapA);
B = zeros(m,n, mapB);
B(:,:) = A;

mapA and mapB are used to create distributed matrices A and B with the following resulting distributions:

Benchmark Platform

Lincoln Laboratory LLGRID
- 80 Node dual 2.8 & 3.06 GHz Xeon (P4)
- 400 & 533 MHz front-side bus
- 4 GB RAM memory per node
- Two 36 GB hard drives per node10/100 Mgmt Ethernet interface
- Two Gig-E Intel interfaces
- 1.02 TB Network Storage
- Running Red Hat Linux

HPCchallenge Benchmark Results: C/MPI vs. pMatlab

	Maximum Problem Size	Execution Performance	Code Size: C/MPI to pMatlab ratio
Random Access	Comparable (128x)	Comparable	6x
Top500	pMatlab (86x), C/MPI (83x)	pMatlab (3x), C/MPI (35x)	66x
FFT	Comparable (128x)	Comparable (55x)	35x
STREAM	Comparable (128x)	Comparable (128x)	8x